```
BBBBBBBBBBBB        AAAAAAAAA     DDDDDDDDDDDD
BBBBBBBBBBBB        AAAAAAAAA     DDDDDDDDDDDD
BBBBBBBBBBBB        AAAAAAAAA     DDDDDDDDDDDD
BBB          BBB   AAA     AAA   DDD          DDD
BBB          BBB   AAA     AAA   DDD          DDD
BBB          BBB   AAA     AAA   DDD          DDD
BBB          BBB   AAA     AAA   DDD          DDD
BBB          BBB   AAA     AAA   DDD          DDD
BBB          BBB   AAA     AAA   DDD          DDD
BBBBBBBBBBBB       AAA     AAA   DDD          DDD
BBBBBBBBBBBB       AAA     AAA   DDD          DDD
BBBBBBBBBBBB       AAA     AAA   DDD          DDD
BBB          BBB   AAAAAAAAAAAAAAA DDD          DDD
BBB          BBB   AAAAAAAAAAAAAAA DDD          DDD
BBB          BBB   AAAAAAAAAAAAAAA DDD          DDD
BBB          BBB   AAA     AAA   DDD          DDD
BBB          BBB   AAA     AAA   DDD          DDD
BBB          BBB   AAA     AAA   DDD          DDD
BBBBBBBBBBBB       AAA     AAA   DDDDDDDDDDDD
BBBBBBBBBBBB       AAA     AAA   DDDDDDDDDDDD
BBBBBBBBBBBB       AAA     AAA   DDDDDDDDDDDD
```

```
BBBBBBBB      AAAAAA    DDDDDDDD     SSSSSSSS  TTTTTTTTTT    AAAAAA     SSSSSSSS  YY      YY    SSSSSSSS
BBBBBBBB      AAAAAA    DDDDDDDD     SSSSSSSS  TTTTTTTTTT    AAAAAA     SSSSSSSS  YY      YY    SSSSSSSS
BB      BB   AA    AA   DD     DD   SS            TT       AA    AA   SS        YY      YY    SS
BB      BB   AA    AA   DD     DD   SS            TT       AA    AA   SS        YY      YY    SS
BB      BB   AA    AA   DD     DD   SS            TT       AA    AA   SS          YY  YY      SS
BB      BB   AA    AA   DD     DD   SS            TT       AA    AA   SS          YY  YY      SS
BBBBBBBB     AA    AA   DD     DD     SSSSSS      TT       AA    AA     SSSSSS      YY        SSSSSS
BBBBBBBB     AA    AA   DD     DD     SSSSSS      TT       AA    AA     SSSSSS      YY        SSSSSS
BB      BB   AAAAAAAAAA DD     DD           SS     TT       AAAAAAAAAA         SS    YY              SS
BB      BB   AAAAAAAAAA DD     DD           SS     TT       AAAAAAAAAA         SS    YY              SS
BB      BB   AA    AA   DD     DD           SS     TT       AA    AA           SS    YY              SS      ....
BB      BB   AA    AA   DD     DD           SS     TT       AA    AA           SS    YY              SS      ....
BBBBBBBB     AA    AA   DDDDDDDD   SSSSSSSS      TT       AA    AA   SSSSSSSS      YY      SSSSSSSS      ....
BBBBBBBB     AA    AA   DDDDDDDD   SSSSSSSS      TT       AA    AA   SSSSSSSS      YY      SSSSSSSS      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;     1        0001  0 MODULE badstasys (%TITLE 'Analyze/Media Stand Alone Support Module'
;     2        0002  0                     IDENT = 'V04-000') =
;     3        0003  1 BEGIN
;     4        0004  1 !
;     5        0005  1 !**************************************************************************
;     6        0006  1 !*                                                                        *
;     7        0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                               *
;     8        0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
;     9        0009  1 !*  ALL RIGHTS RESERVED.                                                   *
;    10        0010  1 !*                                                                        *
;    11        0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    12        0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    13        0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    14        0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    15        0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    16        0016  1 !*  TRANSFERRED.                                                           *
;    17        0017  1 !*                                                                        *
;    18        0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    19        0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    20        0020  1 !*  CORPORATION.                                                           *
;    21        0021  1 !*                                                                        *
;    22        0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    23        0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    24        0024  1 !*                                                                        *
;    25        0025  1 !*                                                                        *
;    26        0026  1 !**************************************************************************
;    27        0027  1 !
;    28        0028  1 !
;    29        0029  1 !++
;    30        0030  1 !
;    31        0031  1 ! Facility:
;    32        0032  1 !
;    33        0033  1 !       Analyze/Media
;    34        0034  1 !
;    35        0035  1 ! Abstract:
;    36        0036  1 !
;    37        0037  1 !       This module contains the routines necessary to support $ ANALYZE/MEDIA
;    38        0038  1 !       (aka BAD) in the standalone environment.
;    39        0039  1 !
;    40        0040  1 ! Environment:
;    41        0041  1 !
;    42        0042  1 !       VAX/VMS User Mode, Non-Privileged
;    43        0043  1 !
;    44        0044  1 ! Author:
;    45        0045  1 !
;    46        0046  1 !       Michael T. Rhodes,              Creation Date: February, 1983
;    47        0047  1 !
;    48        0048  1 ! Modified By:
;    49        0049  1 !
;    50        0050  1 !       V03-001 MTR0006         Michael T. Rhodes               28-Apr-1983
;    51        0051  1 !               Change references to ANALYZECMD to ANALYZCMD for fiche program.
;    52        0052  1 !
;    53        0053  1 !--
;    54        0054  1
```

```
;    56          0055  1 %SBTTL 'Declarations'
;    57          0056  1
;    58          0057  1 !
;    59          0058  1 ! Include Files:
;    60          0059  1 !
;    61          0060  1 REQUIRE 'lib$:baddef';                          ! Define BAD's structures.
;    62          0180  1 LIBRARY 'SYS$LIBRARY:LIB';                      ! Define VMS structures.
;    63          0181  1
;    64          0182  1 !
;    65          0183  1 ! Special Addressing/Linkage Declarations:
;    66          0184  1 !
;    67          0185  1 LINKAGE
;    68          0186  1     JSB = JSB : NOPRESERVE(2,3,4,5,6,7,8,9,10,11),      ! Define JSB linkage for Bugcheck Code.
;    69          0187  1     JSB_R0 = JSB(REGISTER=0): PRESERVE(0,1,2,3,4,5,6,7,8,9,10,11),
;    70          0188  1     JSB_PRESERVE = JSB:,
;    71          0189  1     INI = JSB : PRESERVE(0,1,2,3,4,5,6,7,8,9,10,11);    ! Define linkage for kernel modification.
;    72          0190  1
;    73          0191  1 FORWARD
;    74          0192  1     bugcheck_msg : VECTOR [14, BYTE];               ! Allow forward references to 'Bugcheck' message.
;    75          0193  1
;    76          0194  1 !
;    77          0195  1 ! Table of Contents:
;    78          0196  1 !
;    79          0197  1 FORWARD ROUTINE
;    80          0198  1     bad$bugcheck        : JSB NOVALUE,              ! Bugcheck handler for Stand Alone BAD.
;    81          0199  1     bad$handler,                                    ! Condition Handler for Stand Alone Bad.
;    82          0200  1     bad$install_code,                               ! Installs KERNEL mode code.
;    83          0201  1     bad$putmsg_actrtn,                              ! Action routine for $PUTMSG.
;    84          0202  1     bad$sta_io,                                     ! Performs stand alone terminal IO.
;    85          0203  1     bad$str_trim        : NOVALUE,                  ! Trim the string, removing any trailing CR.
;    86          0204  1     bad$validate_pack   : NOVALUE,                  ! Set Volume Valid bit in device UCB.
;    87          0205  1     LIB$GET_INPUT,                                  ! Library input routine.
;    88          0206  1     SYS$CLOSE,                                      ! Revectored SYSTEM/RMS routines to perform
;    89          0207  1     SYS$CONNECT,                                    ! psuedo file operations on SYS$OUTPUT.
;    90          0208  1     SYS$CREATE,                                     !
;    91          0209  1     SYS$PUT;                                        !
;    92          0210  1
;    93          0211  1 !
;    94          0212  1 ! Private Storage
;    95          0213  1 !
;    96          0214  1 OWN
;    97          0215  1     dyn_buf_adr,                                    ! Starting address of the dynamic string buffer.
;    98          0216  1     tt_chan,                                        ! Channel number
;    99          0217  1     tt_iosb     : VECTOR [4, WORD];                 ! IO status block
;   100          0218  1
;   101          0219  1 BIND                                                !
;   102          0220  1     sys$output =        $DESCRIPTOR ('SYS$OUTPUT'),     ! Terminal Name.
;   103          0221  1     image_name =        $DESCRIPTOR ('STABACKUP.EXE');  ! Image name for stand alone restart.
;   104          0222  1
;   105          0223  1 !
;   106          0224  1 ! External references:
;   107          0225  1 !
;   108          0226  1 EXTERNAL ROUTINE
;   109          0227  1     bad$alloc_mem       : ADDRESSING_MODE (GENERAL),    ! Allocate virtual memory.
;   110          0228  1     bad$sync_io         : ADDRESSING_MODE (GENERAL),    ! Perform synchronous IO.
;   111          0229  1     boo$actimage        : ADDRESSING_MODE (GENERAL),    ! Restart the stand alone image.
;   112          0230  1     ini$rdonly          : INI NOVALUE ADDRESSING_MODE (GENERAL),! Make VMS kernel read only.
```

```
;   113     0231  1     ini$writable        : INI NOVALUE ADDRESSING_MODE (GENERAL),! Make VMS kernel writeable.
;   114     0232  1     CLI$DCL_PARSE       : ADDRESSING_MODE (GENERAL),    ! CLI call back routine.
;   115     0233  1     CLI$GET_VALUE       : ADDRESSING_MODE (GENERAL),    ! CLI call back routine.
;   116     0234  1     CLI$PRESENT         : ADDRESSING_MODE (GENERAL),    ! CLI call back routine.
;   117     0235  1     STR$TRIM            : ADDRESSING_MODE (GENERAL),    ! String routine to trim trailing blanks and tabs.
;   118     0236  1     CON$PUTCHAR         : ADDRESSING_MODE(GENERAL) JSB_R0 NOVALUE,! Put a character out to the console termi
;   119     0237  1     CON$OWNCTY          : ADDRESSING_MODE(GENERAL) JSB_PRESERVE NOVALUE;! "Allocate" the console terminal
;   120     0238  1
;   121     0239  1 EXTERNAL
;   122     0240  1     exe$bug_check       : ADDRESSING_MODE (GENERAL),    ! Address of the bugcheck code.
;   123     0241  1     analyzcmd,                                          ! Command Language Definition module.
;   124     0242  1     bad$gl_chan,                                        ! IO channel.
;   125     0243  1     bad$ga_comnd_line   : $BBLOCK [dsc$c_s_bln],        ! Command line descriptor.
;   126     0244  1     bad$gl_context      : BITVECTOR [32],               ! Context bits.
;   127     0245  1     bad$ga_device       : $BBLOCK [dsc$c_s_bln],        ! Device name descriptor.
;   128     0246  1     bad$gq_devnam       : $BBLOCK [dsc$c_s_bln],        ! Device name descriptor.
;   129     0247  1     bad$ga_filespec     : $BBLOCK [dsc$c_s_bln],        ! File name descriptor.
;   130     0248  1     bad$gl_func,                                        ! IO function.
;   131     0249  1     bad$ga_input_desc   : $BBLOCK [dsc$c_s_bln],        ! Generic input descriptor.
;   132     0250  1     bad$gl_maxblock,                                    ! Total number of blocks on the device.
;   133     0251  1     bad$ga_mdbsf        : REF $BBLOCK,                  ! Address of the MDBSF.
;   134     0252  1     bad$gl_pagcnt,                                      ! Number of pages in each data buffer (1 & 2).
;   135     0253  1     bad$ga_sdbsf        : REF $BBLOCK,                  ! Address of the SDBSF.
;   136     0254  1     bad$gl_serialnum,                                   ! Serial number for device.
;   137     0255  1     bad$gl_status;                                      ! Global status.
;   138     0256  1
;   139     0257  1 !
;   140     0258  1 ! Define message codes...
;   141     0259  1 !
;   142     0260  1 EXTERNAL LITERAL
;   143     0261  1     bad$_closeout,                                      ! Error closing output.
;   144     0262  1     bad$_openout,                                       ! Error opening output file.
;   145     0263  1     bad$_readerr,                                       ! Read error.
;   146     0264  1     bad$_writeerr;                                      ! Write error.
;   147     0265  1
```

```
;  149          0266  1 GLOBAL ROUTINE bad$sta_init : NOVALUE =
;  150          0267  1 !++
;  151          0268  1 !
;  152          0269  1 ! Functional Description:
;  153          0270  1 !
;  154          0271  1 !    This procedure is responsible for performing all of the necessary
;  155          0272  1 !    initialization for $ ANALYZE/MEDIA in the Stand Alone Environment.
;  156          0273  1 !    This includes such things as replacing the BUGCHECK code with a crude
;  157          0274  1 !    local bugcheck handler, allocation of dynamic buffers for the various
;  158          0275  1 !    dynamic string descriptors, prompting for a command line and performing
;  159          0276  1 !    the initial command parse.
;  160          0277  1 !
;  161          0278  1 ! Implicit Outputs:
;  162          0279  1 !
;  163          0280  1 !    The BUGCHECK code has been installed, the dynamic string descriptors
;  164          0281  1 !    have dynamic buffers allocated for them (and have been initialized),
;  165          0282  1 !    the command line has been obtained and parsed.
;  166          0283  1 !
;  167          0284  1 !--
;  168          0285  2 BEGIN
;  169          0286  2
;  170          0287  2 IF .dyn_buf_adr EQL 0                                   ! If this is the first invocation, install the bugch
;  171          0288  2 THEN                                                    ! routine, allocate the required data buffers and
;  172          0289  3     BEGIN                                              ! initialize the descriptors.
;  173          0290  3     $CMKRNL (ROUTIN=bad$install_code);                 ! Install the bugcheck handler.
;  174          0291  3     bad$alloc_mem (bad$k_page_size, dyn_buf_adr);      ! Allocate the dynamic string buffers.
;  175          0292  3     END
;  176          0293  2 ELSE                                                    ! Its not the first invocation, therefore we must cl
;  177        P 0294  2     $CNTREG (PAGCNT =                                  ! the buffers which were previously allocated.
;  178        P 0295  2                     (IF .bad$gl_context [ctx_v_ltdevice]! If the device is a last track format device then
;  179        P 0296  2                     THEN 2                             ! the there are 2 buffers used for the MDBSF and SDB
;  180        P 0297  2                     ELSE 1) +                          ! otherwise there is just one buffer used for the SD
;  181        P 0298  2                     (IF .bad$gl_context [ctx_v_exercise]! If the device was exercised then deallocate the da
;  182        P 0299  2                     THEN .bad$gl_pagcnt * 2 + 1        ! buffers and the test pattern buffer.
;  183          0300  2                     ELSE 0));
;  184          0301  2
;  185          0302  2 bad$ga_comnd_line [dsc$w_length] = bad$k_command_len;   ! Set command line buffer length.
;  186          0303  2 bad$ga_comnd_line [dsc$b_dtype] = dsc$k_dtype_t;        ! Data type is text.
;  187          0304  2 bad$ga_comnd_line [dsc$b_class] = dsc$k_class_s;       ! Class is fixed length string.
;  188          0305  2 bad$ga_comnd_line [dsc$a_pointer] = .dyn_buf_adr;      ! Set the starting address of the dynamic buffer.
;  189          0306  2
;  190          0307  2 bad$ga_device [dsc$w_length] = bad$k_devnam_len;       ! Device specification descriptor.
;  191          0308  2 bad$ga_device [dsc$b_dtype] = dsc$k_dtype_t;           !
;  192          0309  2 bad$ga_device [dsc$b_class] = dsc$k_class_s;           !
;  193          0310  2 bad$ga_device [dsc$a_pointer] = .bad$ga_comnd_line [dsc$a_pointer] + bad$k_command_len;
;  194          0311  2
;  195          0312  2 bad$ga_input_desc [dsc$w_length] = bad$k_input_len;     ! General purpose input descriptor.
;  196          0313  2 bad$ga_input_desc [dsc$b_dtype] = dsc$k_dtype_t;        !
;  197          0314  2 bad$ga_input_desc [dsc$b_class] = dsc$k_class_s;       !
;  198          0315  2 bad$ga_input_desc [dsc$a_pointer] = .bad$ga_device [dsc$a_pointer] + bad$k_devnam_len;
;  199          0316  2
;  200          0317  2 bad$ga_filespec [dsc$w_length] = bad$k_filename_len;   ! Output file specification descriptor.
;  201          0318  2 bad$ga_filespec [dsc$b_dtype] = dsc$k_dtype_t;        !
;  202          0319  2 bad$ga_filespec [dsc$b_class] = dsc$k_class_s;        !
;  203          0320  2 bad$ga_filespec [dsc$a_pointer] = .bad$ga_input_desc [dsc$a_pointer] + bad$k_input_len;
;  204          0321  2
;  205          0322  3 DO  BEGIN                                              ! Get a command line and validate it!
```

```
;  206          0323  3     LIB$GET_INPUT (bad$ga_input_desc, $descriptor (%CHAR(%X'0D',%X'0A'),'$ '));
;  207          0324  3     bad$gl_status = CLI$DCL_PARSE (bad$ga_input_desc, analyzcmd);
;  208          0325  3     END
;  209          0326  2 UNTIL .bad$gl_status;                                    ! If the command syntax is ok, Return.
;  210          0327  2
;  211          0328  1 END;    ! of GLOBAL ROUTINE bad$sta_init


                                                        .TITLE  BADSTASYS Analyze/Media Stand Alone Support Mod
;                                                                               ule
                                                        .IDENT  \V04-000\

                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                54  55  50  54  55  4F  24  53  59  53  00000 P.AAB:  .ASCII  \SYS$OUTPUT\                          ;
                                                        0000A           .BLKB   2
                                            0000000A    0000C P.AAA:  .LONG   10                                  ;
                                            00000000'   00010         .ADDRESS P.AAB                            ;
        45  58  45  2E  50  55  4B  43  41  42  41  54  53  00014 P.AAD:  .ASCII  \STABACKUP.EXE\                   ;
                                                        00021           .BLKB   3
                                            0000000D    00024 P.AAC:  .LONG   13                                  ;
                                            00000000'   00028         .ADDRESS P.AAD                            ;
                                              0A  0D    0002C P.AAF:  .ASCII  <13><10>                            ;
                                              20  24    0002E         .ASCII  \$ \                              ;
                                            00000004    00030 P.AAE:  .LONG   4                                   ;
                                            00000000'   00034         .ADDRESS P.AAF                            ;

                                                        .PSECT  $OWN$,NOEXE,2

                                                    00000 DYN_BUF_ADR:
                                                                .BLKB   4
                                                    00004 TT_CHAN:.BLKB   4
                                                    00008 TT_IOSB:.BLKB   8

                                                        SYS$OUTPUT=         P.AAA
                                                        IMAGE_NAME=         P.AAC
                                                                .EXTRN  BAD$ALLOC_MEM, BAD$SYNC_IO
                                                                .EXTRN  BOO$ACTIMAGE, INI$RDONLY
                                                                .EXTRN  INI$WRITABLE, CLI$DCL_PARSE
                                                                .EXTRN  CLI$GET_VALUE, CLI$PRESENT
                                                                .EXTRN  STR$TRIM, CON$PUTCHAR
                                                                .EXTRN  CON$OWNCTY, EXE$BUG_CHECK
                                                                .EXTRN  ANALYZCMD, BAD$GL_CHAN
                                                                .EXTRN  BAD$GA_COMND_LINE
                                                                .EXTRN  BAD$GL_CONTEXT, BAD$GA_DEVICE
                                                                .EXTRN  BAD$GQ_DEVNAM, BAD$GA_FILESPEC
                                                                .EXTRN  BAD$GL_FUNC, BAD$GA_INPUT_DESC
                                                                .EXTRN  BAD$GL_MAXBLOCK
                                                                .EXTRN  BAD$GA_MDBSF, BAD$GL_PAGCNT
                                                                .EXTRN  BAD$GA_SDBSF, BAD$GL_SERIALNUM
                                                                .EXTRN  BAD$GL_STATUS, BAD$_CLOSEOUT
                                                                .EXTRN  BAD$_OPENOUT, BAD$_READERR
                                                                .EXTRN  BAD$_WRITEERR, SYS$CMKRNL
                                                                .EXTRN  SYS$CNTREG

                                                                .PSECT  $CODE$,NOWRT,2
```

```
                                 000C 00000          .ENTRY  BAD$STA_INIT, Save R2,R3                        ; 0266
                          53     0000'  CF   9E 00002        MOVAB   DYN_BUF_ADR, R3                         ;
                          52     0000G  CF   9E 00007        MOVAB   BAD$GA_INPUT_DESC, R2                   ;
                                        63   D5 0000C        TSTL    DYN_BUF_ADR                             ; 0287
                                        1D   12 0000E        BNEQ    1$                                      ;
                                        7E   D4 00010        CLRL    -(SP)                                   ; 0290
                                0000V   CF   9F 00012        PUSHAB  BAD$INSTALL_CODE                        ;
                00000000G       00           02   FB 00016  CALLS   #2, SYS$CMKRNL                          ;
                                             53   DD 0001D  PUSHL   R3                                       ; 0291
                          7E    0200   8F   3C 0001F        MOVZWL  #512, -(SP)                             ;
                00000000G       00           02   FB 00024  CALLS   #2, BAD$ALLOC_MEM                       ;
                                             2F   11 0002B  BRB     6$                                      ; 0287
                                             7E   7C 0002D  1$:     CLRQ    -(SP)                           ; 0300
                                             7E   D4 0002F          CLRL    -(SP)                           ;
                          05     0000G  CF   E9 00031        BLBC    BAD$GL_CONTEXT+1, 2$                    ;
                          51           02   D0 00036        MOVL    #2, R1                                  ;
                                       03   11 00039        BRB     3$                                      ;
                          51           01   D0 0003B  2$:     MOVL    #1, R1                                  ;
                    0C    0000G  CF    02   E1 0003E  3$:     BBC     #2, BAD$GL_CONTEXT, 4$                  ;
                          50     0000G  CF   D0 00044        MOVL    BAD$GL_PAGCNT, R0                       ;
                          50           02   C4 00049        MULL2   #2, R0                                  ;
                                       50   D6 0004C        INCL    R0                                      ;
                                       02   11 0004E        BRB     5$                                      ;
                                       50   D4 00050  4$:     CLRL    R0                                      ;
                                     6041   9F 00052  5$:     PUSHAB  (R0)[R1]                                ;
                00000000G       00           04   FB 00055  CALLS   #4, SYS$CNTREG                          ;
                     0000G      CF 010E0084  8F   D0 0005C  6$:     MOVL    #17694852, BAD$GA_COMND_LINE    ; 0302
                     0000G      CF           63   D0 00065        MOVL    DYN_BUF_ADR, BAD$GA_COMND_LINE+4    ; 0305
                     0000G      CF 010E003F  8F   D0 0006A        MOVL    #17694783, BAD$GA_DEVICE            ; 0307
          0000G     CF         0000G  CF 00000084  8F   C1 00073  ADDL3   #132, BAD$GA_COMND_LINE+4, BAD$GA_DEVICE+4 ; 0310
                                       62 010E0084  8F   D0 0007F  MOVL    #17694852, BAD$GA_INPUT_DESC        ; 0312
            04    A2         0000G  CF            3F   C1 00086  ADDL3   #63, BAD$GA_DEVICE+4, BAD$GA_INPUT_DESC+4 ; 0315
                             0000G  CF 010E0084  8F   D0 0008D  MOVL    #17694852, BAD$GA_FILESPEC          ; 0317
          0000G     CF           04   A2 00000084  8F   C1 00096  ADDL3   #132, BAD$GA_INPUT_DESC+4, -       ; 0320
                                                                        BAD$GA_FILESPEC+4                   ;
                                             0000'  CF   9F 000A1  7$:     PUSHAB  P.AAE                       ; 0323
                                             52   DD 000A5        PUSHL   R2                                  ;
                                     0000V   CF           02   FB 000A7  CALLS   #2, LIB$GET_INPUT            ;
                                             0000G  CF   9F 000AC        PUSHAB  ANALYZCMD                   ; 0324
                                             52   DD 000B0        PUSHL   R2                                  ;
                        00000000G       00           02   FB 000B2  CALLS   #2, CLI$DCL_PARSE               ;
                             0000G      CF           50   D0 000B9  MOVL    R0, BAD$GL_STATUS               ;
                                        DE   0000G  CF   E9 000BE  BLBC    BAD$GL_STATUS, 7$                   ; 0326
                                                    04   000C3    RET                                         ; 0328

; Routine Size:  196 bytes,    Routine Base:  $CODE$ + 0000

;  212          0329  1
```

```
;  214          0330  1 %SBTTL  'bad$validate_pack -- Set Volume Valid bit in device UCB'
;  215          0331  1 GLOBAL ROUTINE bad$validate_pack : NOVALUE =
;  216          0332  1 !++
;  217          0333  1 !
;  218          0334  1 ! Functional Description:
;  219          0335  1 !
;  220          0336  1 !    Since there is no DCL (hence, no $ MOUNT command) in the Stand Alone
;  221          0337  1 !    Environment, we must manually set the Volume Valid bit in the device
;  222          0338  1 !    UCB.  This assures the IO subsystem that a pack exists in the drive
;  223          0339  1 !    and is spun up ready for action.
;  224          0340  1 !
;  225          0341  1 ! Implicit Outputs:
;  226          0342  1 !
;  227          0343  1 !    The Volume Valid bit has been set.
;  228          0344  1 !
;  229          0345  1 !--
;  230          0346  2 BEGIN
;  231          0347  2 bad$gl_func = IO$_PACKACK;                          ! Set the Volume Valid Bit in the UCB.
;  232          0348  2 DO bad$gl_status = bad$sync_io () UNTIL .bad$gl_status; ! Wait for Drive Ready before continuing.
;  233          0349  1 END;    ! of GLOBAL ROUTINE bad$validate_pack



                                        0000 00000          .ENTRY  BAD$VALIDATE_PACK, Save nothing         ; 0331
                    0000G  CF               08  D0 00002    MOVL    #8, BAD$GL_FUNC                         ; 0347
                00000000G  00               00  FB 00007 1$:  CALLS   #0, BAD$SYNC_IO                         ; 0348
                    0000G  CF               50  D0 0000E    MOVL    R0, BAD$GL_STATUS                       ;
                           EF     0000G     CF  E9 00013    BLBC    BAD$GL_STATUS, 1$                       ;
                                            04 00018        RET                                             ; 0349

; Routine Size:  25 bytes,    Routine Base:  $CODE$ + 00C4

;  234          0350  1
```

```
;  236     0351  1 %SBTTL  'bad$str_trim -- Trim string of trailing blanks, tabs and CR'
;  237     0352  1 GLOBAL ROUTINE bad$str_trim (string_desc) : NOVALUE =
;  238     0353  1 !++
;  239     0354  1 !
;  240     0355  1 ! Functional Description:
;  241     0356  1 !
;  242     0357  1 !    Trim the string provided using the string routine STR$TRIM.
;  243     0358  1 !    Once the string has had the trailing blanks and tabs removed check
;  244     0359  1 !    to see if a trailing carriage return is on the line, if it is, then
;  245     0360  1 !    decrement the string count by one.
;  246     0361  1 !
;  247     0362  1 ! Inputs:
;  248     0363  1 !
;  249     0364  1 !    string_desc        adr     Address of the string descriptor.
;  250     0365  1 !
;  251     0366  1 ! Ouputs:
;  252     0367  1 !
;  253     0368  1 !    string_desc        adr     Address of the string descriptor.
;  254     0369  1 !
;  255     0370  1 ! Side Effects:
;  256     0371  1 !
;  257     0372  1 !    If the string has a carriage return as its last character, it is
;  258     0373  1 !    removed by decrementing the length.
;  259     0374  1 !
;  260     0375  1 !--
;  261     0376  2 BEGIN
;  262     0377  2
;  263     0378  2 LITERAL
;  264     0379  2     cr = %X'0D';
;  265     0380  2
;  266     0381  2 LOCAL
;  267     0382  2     char : BYTE;
;  268     0383  2
;  269     0384  2 MAP
;  270     0385  2     string_desc : REF $BBLOCK;
;  271     0386  2
;  272     0387  2 STR$TRIM (string_desc [dsc$w_length],
;  273     0388  2           string_desc [dsc$w_length],
;  274     0389  2           string_desc [dsc$w_length]);
;  275     0390  2
;  276     0391  3 char = .(.string_desc [dsc$a_pointer] +
;  277     0392  2         (.string_desc [dsc$w_length] - 1));
;  278     0393  2
;  279     0394  2 IF .char EQL cr
;  280     0395  2 THEN string_desc [dsc$w_length] = .string_desc [dsc$w_length] - 1;
;  281     0396  2
;  282     0397  1 END;     ! of GLOBAL ROUTINE bad$str_trim


                                    0004 00000          .ENTRY  BAD$STR_TRIM, Save R2                                   ; 0352
                          52    04  AC  DD 00002          MOVL    STRING_DESC, R2                                         ; 0389
                                    52  DD 00006          PUSHL   R2
                                    52  DD 00008          PUSHL   R2
                                    52  DD 0000A          PUSHL   R2
```

```
                    00000000G  00          03  FB 0000C          CALLS   #3, STR$TRIM                          ;
                               50          62  3C 00013          MOVZWL  (R2), R0                            ; 0392
                               50      04  A2  C0 00016          ADDL2   4(R2), R0                           ;
                               51      FF  A0  90 0001A          MOVB    -1(R0), CHAR                        ; 0391
                               0D          51  91 0001E          CMPB    CHAR, #13                           ; 0394
                                           02  12 00021          BNEQ    1$                                  ;
                                           62  B7 00023          DECW    (R2)                                ; 0395
                                               04 00025 1$:      RET                                         ; 0397

; Routine Size:  38 bytes,    Routine Base:  $CODE$ + 00DD

;  283          0398  1
```

```
;   285          0399  1 %SBTTL  'Revectored Library Routine -- LIB$GET_INPUT'
;   286          0400  1 GLOBAL ROUTINE LIB$GET_INPUT (desc, prompt, retlen) =
;   287          0401  1 !++
;   288          0402  1 !
;   289          0403  1 ! Functional Description:
;   290          0404  1 !
;   291          0405  1 !    Stand Alone Environment replacement module, which simply calls a local
;   292          0406  1 !    support procedure to perform a $QIOW with a read with prompt option.
;   293          0407  1 !
;   294          0408  1 ! Inputs:
;   295          0409  1 !
;   296          0410  1 !    desc        adr     address of the input buffer descriptor.
;   297          0411  1 !    prompt      adr     address of the prompt string descriptor.
;   298          0412  1 !    retlen      adr     address of a word to receive the actual string length.
;   299          0413  1 !
;   300          0414  1 ! Outputs:
;   301          0415  1 !
;   302          0416  1 !    The input string has been read and its contents reside in the buffer
;   303          0417  1 !    pointed to by 'desc'.  The length field has been updated in the descriptor.
;   304          0418  1 !
;   305          0419  1 ! Side Effects:
;   306          0420  1 !
;   307          0421  1 !    The buffer is initialized to all spaces prior to reading the input line.
;   308          0422  1 !
;   309          0423  1 !--
;   310          0424  2 BEGIN
;   311          0425  2 BUILTIN
;   312          0426  2     NULLPARAMETER;
;   313          0427  2
;   314          0428  2 MAP
;   315          0429  2     desc : REF $BBLOCK;
;   316          0430  2
;   317          0431  2 CH$FILL (%C' ', .desc [dsc$w_length], .desc [dsc$a_pointer]);
;   318          0432  2
;   319          0433  3 RETURN (bad$sta_io (IO$_READPROMPT,
;   320          0434  3                     desc [dsc$w_length],
;   321          0435  3                     .prompt,
;   322          0436  3                     IF NOT NULLPARAMETER (3)
;   323          0437  3                     THEN .retlen
;   324          0438  2                     ELSE 0));
;   325          0439  2
;   326          0440  1 END;    ! of GLOBAL ROUTINE LIB$GET_INPUT
```

```
                                 007C 00000     .ENTRY  LIB$GET_INPUT, Save R2,R3,R4,R5,R6      ; 0400
                          56     04 AC  D0 00002 MOVL    DESC, R6                                ; 0431
                20        6E        00  2C 00006 MOVC5   #0, (SP), #32, (R6), @4(R6)
                                    04  B6 0000B
                          03            6C  91 0000D CMPB    (AP), #3                          ; 0436
                                        0A  1F 00010 BLSSU   1$
                                     0C AC  D5 00012 TSTL    12(AP)
                                        05  13 00015 BEQL    1$
                                     0C AC  DD 00017 PUSHL   RETLEN                            ; 0437
                                        02  11 0001A BRB     2$
```

```
                                 7E  D4 0001C 1$:     CLRL    -(SP)                                   ; 0436
                              08 AC  DD 0001E 2$:     PUSHL   PROMPT                                  ; 0435
                                 56  DD 00021         PUSHL   R6                                      ; 0434
                                 37  DD 00023         PUSHL   #55                                     ; 0433
                      0000V  CF  04  FB 00025         CALLS   #4, BAD$STA_IO                          ;
                                 04  0002A           RET                                             ; 0440

; Routine Size:  43 bytes,    Routine Base:  $CODE$ + 0103

;  327          0441  1
```

```
;   329          0442  1 %SBTTL  'SYS$CREATE -- Custom Stand Alone support procedure'
;   330          0443  1 GLOBAL ROUTINE SYS$CREATE (fab, err, suc) =
;   331          0444  1 !++
;   332          0445  1 !
;   333          0446  1 ! Functional Description:
;   334          0447  1 !
;   335          0448  1 !    In the Stand Alone environment RMS does not exist, hence this routine
;   336          0449  1 !    provides a crude solution for performing the $CREATE actions.  The input
;   337          0450  1 !    arguments supplied by the caller are IGNORED.  The following actions are
;   338          0451  1 !    performed:  translate the logical name 'SYS$OUTPUT' provided by the system
;   339          0452  1 !    and assign a channel to the device.
;   340          0453  1 !
;   341          0454  1 ! Inputs:
;   342          0455  1 !
;   343          0456  1 !    fab         adr     address of the FAB (ignored)
;   344          0457  1 !    err         adr     address of an error action routine (ignored)
;   345          0458  1 !    suc         adr     address of a success action routine (ignored)
;   346          0459  1 !
;   347          0460  1 ! Implicit Outputs:
;   348          0461  1 !
;   349          0462  1 !    A channel is established to SYS$OUTPUT.
;   350          0463  1 !
;   351          0464  1 !--
;   352          0465  2 BEGIN
;   353          0466  2 LOCAL
;   354          0467  2     tt_desc     : $BBLOCK [dsc$c_s_bln],                ! Device name descriptor
;   355          0468  2     tt_name     : $BBLOCK [bad$k_devnam_len + 1];       ! Device name buffer
;   356          0469  2
;   357          0470  2 IF .tt_chan EQL 0                                      ! Has a channel already been established?
;   358          0471  2 THEN                                                   ! No, allocate a CCB to point to SYS$OUTPUT.
;   359          0472  3     BEGIN                                               !
;   360          0473  3     CH$FILL (0, dsc$c_s_bln, tt_desc);                  ! Initialize the descriptor.
;   361          0474  3     tt_desc [dsc$w_length] = bad$k_devnam_len + 1;      ! Length
;   362          0475  3     tt_desc [dsc$a_pointer] = tt_name;                  ! Address
;   363          0476  3
;   364        P 0477  3     $TRNLOG (LOGNAM = sys$output,                       ! Translate SYS$OUTPUT to determine the
;   365        P 0478  3              RSLLEN = tt_desc,                          ! correct output device.
;   366          0479  3              RSLBUF = tt_desc);
;   367          0480  3
;   368          0481  3     $ASSIGN (DEVNAM = tt_desc, CHAN = tt_chan);         ! Assign a channel to the output device.
;   369          0482  2     END;
;   370          0483  2
;   371          0484  2 RETURN ss$_normal;
;   372          0485  2
;   373          0486  1 END;    ! of GLOBAL ROUTINE SYS$CREATE


                                                        .EXTRN  SYS$TRNLOG, SYS$ASSIGN

                                                003C 00000      .ENTRY  SYS$CREATE, Save R2,R3,R4,R5            ; 0443
                                5E      B8  AE  9E 00002        MOVAB   -72(SP), SP                             ;
                              0000'     CF  D5 00006            TSTL    TT_CHAN                                 ; 0470
                                        35  12 0000A            BNEQ    1$                                      ;
        08              00        6E      00  2C 0000C        MOVC5   #0, (SP), #0, #8, TT_DESC               ; 0473
                                              40  AE 00011
                                40  AE      40  8F  9B 00013    MOVZBW  #64, TT_DESC                            ; 0474
```

```
                          44   AE              6E  9E 00018         MOVAB   TT_NAME, TT_DESC+4                      ; 0475
                                               7E  7C 0001C         CLRQ    -(SP)                                  ; 0479
                                               7E  D4 0001E         CLRL    -(SP)                                  ;
                                          4C   AE  9F 00020         PUSHAB  TT_DESC                                ;
                                          50   AE  9F 00023         PUSHAB  TT_DESC                                ;
                                        0000'  CF  9F 00026         PUSHAB  SYS$OUTPUT                             ;
                  00000000G  00                 06  FB 0002A         CALLS   #6, SYS$TRNLOG                         ;
                                               7E  7C 00031         CLRQ    -(SP)                                  ; 0481
                                        0000'  CF  9F 00033         PUSHAB  TT_CHAN                                ;
                                          4C   AE  9F 00037         PUSHAB  TT_DESC                                ;
                  00000000G  00                 04  FB 0003A         CALLS   #4, SYS$ASSIGN                         ;
                             50                 01  D0 00041 1$:     MOVL    #1, R0                                 ; 0484
                                                04  00044           RET                                            ; 0486

; Routine Size:  69 bytes,    Routine Base:  $CODE$ + 012E

;  374          0487  1
```

```
;  376          0488  1 %SBTTL  'SYS$CONNECT -- Custom Stand Alone support procedure'
;  377          0489  1 GLOBAL ROUTINE SYS$CONNECT (rab, err, suc) =
;  378          0490  1 !++
;  379          0491  1 !
;  380          0492  1 ! Functional Description:
;  381          0493  1 !
;  382          0494  1 !     In the Stand Alone environment RMS does not exist, hence this routine
;  383          0495  1 !     provides a crude solution for performing the $CONNECT actions.  The input
;  384          0496  1 !     arguments are IGNORED.  This routine performs only one (1) action: it
;  385          0497  1 !     returns to the caller with sucess status.
;  386          0498  1 !
;  387          0499  1 !--
;  388          0500  2 BEGIN
;  389          0501  2 RETURN ss$_normal;
;  390          0502  1 END;    ! of GLOBAL ROUTINE SYS$CONNECT



                                        0000 00000          .ENTRY  SYS$CONNECT, Save nothing                       ; 0489
                                50      01  D0 00002          MOVL    #1, R0                                          ; 0501
                                        04 00005            RET                                                     ; 0502

; Routine Size:  6 bytes,    Routine Base:  $CODE$ + 0173

;  391          0503  1
```

```
;  393          0504  1 %SBTTL  'SYS$PUT -- Custom Stand Alone support procedure'
;  394          0505  1 GLOBAL ROUTINE SYS$PUT (rab, err, suc) =
;  395          0506  1 !++
;  396          0507  1 !
;  397          0508  1 ! Functional Description:
;  398          0509  1 !
;  399          0510  1 !    In the Stand Alone environment RMS does not exist, hence this routine
;  400          0511  1 !    provides a crude solution for performing the $PUT actions.  The 'rab'
;  401          0512  1 !    input argument is the only argument used, the others are IGNORED.
;  402          0513  1 !
;  403          0514  1 !    The action taken is as follows:
;  404          0515  1 !
;  405          0516  1 !       The buffer address and size are extracted from the RAB, and placed
;  406          0517  1 !       into a local descriptor.  The size of the line is checked for a
;  407          0518  1 !       length of zero (0) indicating a blank line should be output.  If
;  408          0519  1 !       this is true, then the descriptor is pointed the a local CRLF pair
;  409          0520  1 !       and the length is set to 2 bytes.  Otherwise, a CRLF is appended
;  410          0521  1 !       to the string passed (fake RAT=CR) and the length is bumped up by
;  411          0522  1 !       2 bytes.  The string is the printed and we return status from the
;  412          0523  1 !       $QIO.
;  413          0524  1 !
;  414          0525  1 ! Inputs:
;  415          0526  1 !
;  416          0527  1 !    rab         adr     address of the RAB
;  417          0528  1 !    err         adr     address of an error routine (ignored)
;  418          0529  1 !    suc         adr     address of a success routine (ignored)
;  419          0530  1 !
;  420          0531  1 ! Implicit Outputs:
;  421          0532  1 !
;  422          0533  1 !    The line has been written to the output file.
;  423          0534  1 !
;  424          0535  1 !--
;  425          0536  2 BEGIN
;  426          0537  2 BUILTIN
;  427          0538  2     PROBEW;
;  428          0539  2
;  429          0540  2 MAP
;  430          0541  2     rab : REF $BBLOCK;
;  431          0542  2
;  432          0543  2 LOCAL
;  433          0544  2     cr_lf : WORD INITIAL (%X'0D0A'),                        ! Local CRLR datum.
;  434          0545  2     line_desc : $BBLOCK [dsc$c_s_bln],                      ! Descriptor formed from the RAB RBF
;  435          0546  2     mode : BYTE INITIAL (prt$c_uw);                         ! Hardware Protection code for User
;  436          0547  2
;  437          0548  2 line_desc [dsc$w_length] = .rab [rab$w_rsz];               ! Get the record size.
;  438          0549  2 line_desc [dsc$a_pointer]= .rab [rab$l_rbf];               ! Get the record buffer address.
;  439          0550  2
;  440          0551  2 IF .line_desc [dsc$w_length] EQL 0                        ! If no string is provided, then we
;  441          0552  2 THEN                                                      ! write a blank line.
;  442          0553  3     BEGIN
;  443          0554  3     line_desc [dsc$a_pointer] = cr_lf;                    ! Address of the carriage return lin
;  444          0555  3     line_desc [dsc$w_length] = 2;                         ! String length.
;  445          0556  3     END
;  446          0557  2 ELSE
;  447          0558  2     IF PROBEW (mode, line_desc [dsc$w_length], .line_desc [dsc$a_pointer]) ! A string was provided, can a CR
;  448          0559  2     THEN                                                  ! Yes, append the carriage return li
;  449          0560  3         BEGIN                                             ! onto the line and adjust the lengt
```

```
;  450          0561  3          .line_desc [dsc$a_pointer] + .line_desc [dsc$w_length] = .cr_lf;
;  451          0562  3          line_desc [dsc$w_length] = .line_desc [dsc$w_length] + 2;
;  452          0563  3          END;
;  453          0564  3
;  454          0565  3      IF NOT (bad$gl_status = bad$sta_io (IO$_WRITELBLK, line_desc))
;  455          0566  2      THEN
;  456          0567  2          SIGNAL (bad$_writeerr, 1, sys$output, .bad$gl_status);
;  457          0568  2
;  458          0569  2      RETURN ss$_normal;
;  459          0570  2
;  460          0571  1  END;    ! of GLOBAL ROUTINE SYS$PUT



                                                0000 00000          .ENTRY  SYS$PUT, Save nothing                           ; 0505
                                          5E    0C   C2 00002          SUBL2   #12, SP
                                          6E  0D0A   8F   B0 00005          MOVW    #3338, CR_LF                            ; 0536
                                          51         04   90 0000A          MOVB    #4, MODE
                                          50    04   AC   D0 0000D          MOVL    RAB, R0                                 ; 0548
                                    04    AE    22   A0   B0 00011          MOVW    34(R0), LINE_DESC
                                    08    AE    28   A0   D0 00016          MOVL    40(R0), LINE_DESC+4                     ; 0549
                                          04    AE   B5 0001B          TSTW    LINE_DESC                               ; 0551
                                                0A   12 0001E          BNEQ    1$
                                    08    AE          6E   9E 00020          MOVAB   CR_LF, LINE_DESC+4                      ; 0554
                                    04    AE          02   B0 00024          MOVW    #2, LINE_DESC                           ; 0555
                                                17   11 00028          BRB     2$                                      ; 0551
                              08    BE    04    AE          51   0D 0002A 1$:      PROBEW  MODE, LINE_DESC, @LINE_DESC+4           ; 0558
                                                0F   13 00030          BEQL    2$
                                          50    04   AE   3C 00032          MOVZWL  LINE_DESC, R0                           ; 0561
                                          50    08   AE   C0 00036          ADDL2   LINE_DESC+4, R0
                                          60         6E   3C 0003A          MOVZWL  CR_LF, (R0)
                                    04    AE          02   A0 0003D          ADDW2   #2, LINE_DESC                           ; 0562
                                                04    AE   9F 00041 2$:      PUSHAB  LINE_DESC                       ; 0565
                                                20   DD 00044          PUSHL   #32
                                    0000V CF          02   FB 00046          CALLS   #2, BAD$STA_IO
                                    0000G CF          50   D0 0004B          MOVL    R0, BAD$GL_STATUS
                                          17          50   E8 00050          BLBS    R0, 3$
                                                0000G CF   DD 00053          PUSHL   BAD$GL_STATUS                   ; 0567
                                                0000' CF   9F 00057          PUSHAB  SYS$OUTPUT
                                                01   DD 0005B          PUSHL   #1
                                          00000000G   8F   DD 0005D          PUSHL   #BAD$_WRITEERR
                                    00000000G   00         04   FB 00063          CALLS   #4, LIB$SIGNAL
                                          50              01   D0 0006A 3$:      MOVL    #1, R0                          ; 0569
                                                          04   0006D          RET                                     ; 0571

; Routine Size:  110 bytes,    Routine Base:  $CODE$ + 0179

;  461          0572  1
```

```
;  463          0573  1 %SBTTL  'SYS$CLOSE -- Custom Stand Alone support procedure'
;  464          0574  1 GLOBAL ROUTINE SYS$CLOSE (fab, err, suc) =
;  465          0575  1 !++
;  466          0576  1 !
;  467          0577  1 ! Functional Description:
;  468          0578  1 !
;  469          0579  1 !    In the Stand Alone environment RMS does not exist, hence this routine
;  470          0580  1 !    provides a crude solution for performing the $CLOSE actions.  The input
;  471          0581  1 !    arguments are IGNORED.  This procedure disconnects the channel from
;  472          0582  1 !    SYS$OUTPUT.
;  473          0583  1 !
;  474          0584  1 ! Implicit Inputs:
;  475          0585  1 !
;  476          0586  1 !    tt_chan             Longword containing the channel to SYS$OUTPUT.
;  477          0587  1 !
;  478          0588  1 ! Side Effects:
;  479          0589  1 !
;  480          0590  1 !    Any problems in closing the output file will be reported via
;  481          0591  1 !    a SIGNAL_STOP.
;  482          0592  1 !
;  483          0593  1 !--
;  484          0594  2 BEGIN
;  485          0595  2 IF NOT (bad$gl_status = $DASSGN (CHAN = .tt_chan))
;  486          0596  2 THEN
;  487          0597  2     SIGNAL_STOP (bad$_closeout, 1, sys$output, .bad$gl_status);
;  488          0598  2
;  489          0599  2 tt_chan = 0;
;  490          0600  2
;  491          0601  2 RETURN ss$_normal;
;  492          0602  2
;  493          0603  1 END;    ! of GLOBAL ROUTINE SYS$CLOSE


                                                        .EXTRN  SYS$DASSGN

                                        0000 00000      .ENTRY  SYS$CLOSE, Save nothing                         ; 0574
                                0000'   CF   DD 00002   PUSHL   TT_CHAN                                        ; 0595
                    00000000G   00      01   FB 00006   CALLS   #1, SYS$DASSGN                                  ;
                        0000G   CF          50   D0 0000D   MOVL    R0, BAD$GL_STATUS                           ;
                                17          50   E8 00012   BLBS    R0, 1$                                      ;
                                        0000G   CF   DD 00015   PUSHL   BAD$GL_STATUS                           ; 0597
                                        0000'   CF   9F 00019   PUSHAB  SYS$OUTPUT                              ;
                                                01   DD 0001D   PUSHL   #1                                      ;
                                    00000000G   8F   DD 0001F   PUSHL   #BAD$_CLOSEOUT                          ;
                    00000000G   00              04   FB 00025   CALLS   #4, LIB$STOP                           ;
                                        0000'   CF   D4 0002C 1$:  CLRL    TT_CHAN                             ; 0599
                                50              01   D0 00030   MOVL    #1, R0                                ; 0601
                                                04   00033     RET                                             ; 0603

; Routine Size:  52 bytes,    Routine Base:  $CODE$ + 01E7

;  494          0604  1
```

```
;  496          0605  1 %SBTTL  'bad$sta_io -- Stand Alone IO support procedure'
;  497          0606  1 GLOBAL ROUTINE bad$sta_io (func, desc, prompt, length) =
;  498          0607  1 !++
;  499          0608  1 !
;  500          0609  1 ! Functional Description:
;  501          0610  1 !
;  502          0611  1 !    Since there is no RMS support in the Stand Alone environment, this
;  503          0612  1 !    routine handles all terminal IO functions by issuing $QIOWs.
;  504          0613  1 !
;  505          0614  1 ! Inputs:
;  506          0615  1 !
;  507          0616  1 !    func        val     IO operation to perform.
;  508          0617  1 !    desc        adr     Address of the string descriptor.
;  509          0618  1 !    prompt      adr     Address of the prompt string descriptor. [OPTIONAL]
;  510          0619  1 !    length      adr     Address of a word to receive the string length. [OPTIONAL]
;  511          0620  1 !
;  512          0621  1 ! Routine Value:
;  513          0622  1 !
;  514          0623  1 !    The routine value is the completion status of the $QIOW.
;  515          0624  1 !
;  516          0625  1 !    IF the $QIOW was successful but the terminating character was a ^Z,
;  517          0626  1 !    then we will return RMS$_EOF.
;  518          0627  1 !
;  519          0628  1 ! Side Effects:
;  520          0629  1 !
;  521          0630  1 !    If a channel has not been established to the output device (SYS$OUTPUT),
;  522          0631  1 !    one will be created before attempting the IO operation.
;  523          0632  1 !
;  524          0633  1 !--
;  525          0634  2 BEGIN
;  526          0635  2 BUILTIN
;  527          0636  2     NULLPARAMETER;
;  528          0637  2
;  529          0638  2 LITERAL
;  530          0639  2     ctrl_z = %X'1A';
;  531          0640  2
;  532          0641  2 MAP
;  533          0642  2     desc        : REF $BBLOCK,
;  534          0643  2     prompt      : REF $BBLOCK;
;  535          0644  2
;  536          0645  2 BIND
;  537          0646  2     char = .desc [dsc$a_pointer] : BYTE;                ! Check first character of the input buffer for ^Z.
;  538          0647  2
;  539          0648  2 IF .tt_chan EQL 0                                       ! Has a channel been established to SYS$OUTPUT?
;  540          0649  2 THEN $CREATE (FAB = sys$output);                        ! Calls revectored service procedure with a dummy ar
;  541          0650  2
;  542          0651  3 IF NOT (bad$gl_status =
;  543        P 0652  3                     $QIOW ( FUNC =  .func,            ! Use supplied IO function.
;  544        P 0653  3                             CHAN =  .tt_chan,         ! Direct output to SYS$OUTPUT.
;  545        P 0654  3                             IOSB =  .tt_iosb,         ! Save the status.
;  546        P 0655  3                             P1 =    .desc [dsc$a_pointer],  ! Buffer address.
;  547        P 0656  3                             P2 =    .desc [dsc$w_length],   ! Buffer size.
;  548        P 0657  3                             P5 =    IF NULLPARAMETER (3)    ! If no prompt was supplied, set the default
;  549        P 0658  3                             THEN 0                          ! of nothing, otherwise use the prompt supplied.
;  550        P 0659  3                             ELSE .prompt [dsc$a_pointer],
;  551        P 0660  3                             P6 =    IF NULLPARAMETER (3)    !
;  552        P 0661  3                             THEN 0                          !
```

```
;   553     0662  3                                       ELSE .prompt [dsc$w_length]))
;   554     0663  2 THEN                                                    ! NOTE:  Any problems restart the image!
;   555     0664  2     SIGNAL (IF .func EQL IO$_WRITELBLK                  ! Force FATAL severity, include the other
;   556     0665  2                THEN bad$_writeerr + sts$k_severe        ! arguments for consistency.
;   557     0666  2                ELSE bad$_readerr + sts$k_severe,
;   558     0667  2                1, sys$output, .bad$gl_status);
;   559     0668  2
;   560     0669  2 IF .func NEQ IO$_WRITELBLK                              ! If we have just completed a READ function...
;   561     0670  2 THEN                                                    ! Special case the following:
;   562     0671  2     BEGIN                                               ! See if the first character in the buffer is
;   563     0672  2     IF .char EQL ctrl_z                                 ! a ^Z.  If it is, then use the RMS$_EOF status
;   564     0673  2     THEN bad$gl_status = RMS$_EOF;                      ! to indicate a terminator was the first char seen.
;   565     0674  2     IF NOT NULLPARAMETER (4)                            ! Was a return length requested?
;   566     0675  2     THEN .length = .tt_iosb [1];                        ! Yes, copy number of bytes read from the IOSB.
;   567     0676  2     END;
;   568     0677  2
;   569     0678  2 RETURN .bad$gl_status;
;   570     0679  2
;   571     0680  1 END;    ! of GLOBAL ROUTINE bad$sta_io


                                                        .EXTRN  SYS$QIOW

                                    001C 00000          .ENTRY  BAD$STA_IO, Save R2,R3,R4                       ; 0606
                            54  0000' CF  9E 00002       MOVAB   TT_CHAN, R4                                     ;
                            53  0000G CF  9E 00007       MOVAB   BAD$GL_STATUS, R3                               ;
                            52    08 AC  D0 0000C       MOVL    DESC, R2                                        ; 0646
                                     64  D5 00010       TSTL    TT_CHAN                                         ; 0648
                                     0B  12 00012       BNEQ    1$                                              ;
                                0000' CF  9F 00014       PUSHAB  SYS$OUTPUT                                      ; 0649
                00000000G     00     01  FB 00018       CALLS   #1, SYS$CREATE                                  ;
                              03         6C  91 0001F 1$:    CMPB    (AP), #3                                    ; 0662
                                         05  1F 00022       BLSSU   2$                                          ;
                                    0C   AC  D5 00024       TSTL    12(AP)                                      ;
                                         04  12 00027       BNEQ    3$                                          ;
                                         7E  D4 00029 2$:    CLRL    -(SP)                                       ;
                                         04  11 0002B       BRB     4$                                          ;
                            7E          0C   BC  3C 0002D 3$:    MOVZWL  @PROMPT, -(SP)                          ;
                              03         6C  91 00031 4$:    CMPB    (AP), #3                                    ;
                                         05  1F 00034       BLSSU   5$                                          ;
                                    0C   AC  D5 00036       TSTL    12(AP)                                      ;
                                         04  12 00039       BNEQ    6$                                          ;
                                         7E  D4 0003B 5$:    CLRL    -(SP)                                       ;
                                         07  11 0003D       BRB     7$                                          ;
                            50          0C   AC  D0 0003F 6$:    MOVL    PROMPT, R0                              ;
                                        04   A0  DD 00043       PUSHL   4(R0)                                   ;
                                         7E  7C  00046 7$:    CLRQ    -(SP)                                       ;
                            7E           62  3C  00048       MOVZWL  (R2), -(SP)                                 ;
                                        04   A2  DD 0004B       PUSHL   4(R2)                                   ;
                                         7E  7C  0004E       CLRQ    -(SP)                                       ;
                                        04   A4  DD 00050       PUSHL   TT_IOSB                                 ;
                                        04   AC  DD 00053       PUSHL   FUNC                                    ;
                                         64  DD  00056       PUSHL   TT_CHAN                                 ;
                                         7E  D4  00058       CLRL    -(SP)                                   ;
                00000000G     00         0C  FB  0005A       CALLS   #12, SYS$QIOW                           ;
                              63         50  D0  00061       MOVL    R0, BAD$GL_STATUS                       ;
```

```
                                23              50  E8 00064          BLBS    R0, 10$                                         ;
                                                63  DD 00067          PUSHL   BAD$GL_STATUS                                   ; 0667
                                      0000'     CF  9F 00069          PUSHAB  SYS$OUTPUT                                      ; 0664
                                                01  DD 0006D          PUSHL   #1                                              ;
                                20        04    AC  D1 0006F          CMPL    FUNC, #32                                       ;
                                                08  12 00073          BNEQ    8$                                              ;
                                   00000000G    8F  DD 00075          PUSHL   #BAD$_WRITEERR+4                                ; 0665
                                                06  11 0007B          BRB     9$                                              ;
                                   00000000G    8F  DD 0007D  8$:     PUSHL   #BAD$_READERR+4                                 ; 0666
                      00000000G 00              04  FB 00083  9$:     CALLS   #4, LIB$SIGNAL                                  ; 0664
                                20        04    AC  D1 0008A  10$:    CMPL    FUNC, #32                                       ; 0669
                                                1C  13 0008E          BEQL    12$                                             ;
                                1A        04    B2  91 00090          CMPB    @4(R2), #26                                     ; 0672
                                                07  12 00094          BNEQ    11$                                             ;
                                63 0001827A     8F  D0 00096          MOVL    #98938, BAD$GL_STATUS                           ; 0673
                                04              6C  91 0009D  11$:     CMPB    (AP), #4                                        ; 0674
                                                0A  1F 000A0          BLSSU   12$                                             ;
                                      10        AC  D5 000A2          TSTL    16(AP)                                          ;
                                                05  13 000A5          BEQL    12$                                             ;
                          10    BC        06    A4  3C 000A7          MOVZWL  TT_IOSB+2, @LENGTH                              ; 0675
                                50              63  D0 000AC  12$:     MOVL    BAD$GL_STATUS, R0                               ; 0678
                                                    04 000AF          RET                                                     ; 0680

; Routine Size:  176 bytes,    Routine Base:  $CODE$ + 021B

;  572          0681  1
```

```
;   574          0682  1 %SBTTL  'Condition Handling Procedures'
;   575          0683  1 GLOBAL ROUTINE bad$handler (signal, mechanism) =
;   576          0684  1 !++
;   577          0685  1 !
;   578          0686  1 ! Functional Description:
;   579          0687  1 !
;   580          0688  1 !    This routine provides the mechanism for generating and displaying
;   581          0689  1 !    error messages SIGNALled by BAD.  To generate the messages we call
;   582          0690  1 !    $PUTMSG, however, the messages are displayed by the $PUTMSG action
;   583          0691  1 !    routine 'bad$putmsg_actrtn'.
;   584          0692  1 !
;   585          0693  1 ! Inputs:
;   586          0694  1 !
;   587          0695  1 !    signal     adr     Address of the Standard VMS Signal Array.
;   588          0696  1 !    mechansim  adr     Address of the Standard VMS Mechanism Array.
;   589          0697  1 !
;   590          0698  1 ! Routine Value:
;   591          0699  1 !
;   592          0700  1 !    SS$_CONTINUE
;   593          0701  1 !
;   594          0702  1 ! Side Effects:
;   595          0703  1 !
;   596          0704  1 !    If the severity of the message is FATAL (SEVERE), the image is restarted.
;   597          0705  1 !
;   598          0706  1 !--
;   599          0707  2 BEGIN
;   600          0708  2 MAP
;   601          0709  2     signal      : REF $BBLOCK,                          ! Signal Array
;   602          0710  2     mechanism   : REF $BBLOCK;                          ! Mechansim Array
;   603          0711  2
;   604          0712  2 IF .signal [chf$l_sig_name] NEQ ss$_unwind
;   605          0713  2 THEN
;   606          0714  2     BEGIN
;   607          0715  2     signal [chf$l_sig_args] = .signal [chf$l_sig_args] - 2;! Strip PC and PSL from the argument list.
;   608          0716  2     $PUTMSG (MSGVEC=.signal, ACTRTN=bad$putmsg_actrtn); ! format and 'display' the message.
;   609          0717  2
;   610          0718  2     IF (.signal [chf$l_sig_name] AND sts$m_severity) EQL sts$k_severe
;   611          0719  2     THEN
;   612          0720  3         BEGIN
;   613          0721  3         boo$actimage (image_name);                      ! Restart the image if severity was FATAL.
;   614          0722  3         $exit ();                                       ! If restart fails, then ungracefully exit.
;   615          0723  3         END;
;   616          0724  2     END;
;   617          0725  2
;   618          0726  2 RETURN ss$_continue;                                   ! Continue processing.
;   619          0727  2
;   620          0728  1 END;    ! of GLOBAL ROUTINE bad$handler


                                                        .EXTRN  SYS$PUTMSG, SYS$EXIT

                                                0004 00000      .ENTRY  BAD$HANDLER, Save R2                            ; 0683
                        52      04      AC  D0 00002    MOVL    SIGNAL, R2                                              ; 0712
                00000920 8F     04      A2  D1 00006    CMPL    4(R2), #2336                                            :
                                        2E  13 0000E    BEQL    1$                                                      :
                        62              02  C2 00010    SUBL2   #2, (R2)                                                ; 0715
```

```
                                                7E  7C 00013          CLRQ    -(SP)                                          ; 0716
                                         0000V  CF  9F 00015          PUSHAB  BAD$PUTMSG_ACTRTN                              ;
                                                52  DD 00019          PUSHL   R2                                             ;
                                00000000G   00  04  FB 0001B          CALLS   #4, SYS$PUTMSG                                 ;
                04        04   A2         03  00  ED 00022          CMPZV   #0, #3, 4(R2), #4                              ; 0718
                                                14  12 00028          BNEQ    1$                                             ;
                                         0000'  CF  9F 0002A          PUSHAB  IMAGE_NAME                                     ; 0721
                                00000000G   00  01  FB 0002E          CALLS   #1, BOO$ACTIMAGE                               ;
                                                01  DD 00035          PUSHL   #1                                             ; 0722
                                00000000G   00  01  FB 00037          CALLS   #1, SYS$EXIT                                   ;
                                            50  01  D0 0003E 1$:      MOVL    #1, R0                                         ; 0726
                                                    04 00041          RET                                                    ; 0728

; Routine Size:  66 bytes,    Routine Base:  $CODE$ + 02CB

;  621          0729  1
```

```
;   623          0730  1 GLOBAL ROUTINE bad$putmsg_actrtn (desc) =
;   624          0731  1 !++
;   625          0732  1 !
;   626          0733  1 ! Functional Description:
;   627          0734  1 !
;   628          0735  1 !    This routine simply calls 'bad$sta_io' to write the formatted
;   629          0736  1 !    signal message from $PUTMSG.  We return FALSE to inhibit $PUTMSG
;   630          0737  1 !    from attempting to write the message too (since there is no RMS
;   631          0738  1 !    support in the stand alone environment!).
;   632          0739  1 !
;   633          0740  1 ! Inputs:
;   634          0741  1 !
;   635          0742  1 !    desc        adr     Address of the message string descriptor.
;   636          0743  1 !
;   637          0744  1 ! Routine Value:
;   638          0745  1 !
;   639          0746  1 !    FALSE       Inhibit $PUTMSG from attempting to write the
;   640          0747  1 !                signal message also.
;   641          0748  1 !
;   642          0749  1 !--
;   643          0750  2 BEGIN
;   644          0751  2 BUILTIN
;   645          0752  2     PROBEW;
;   646          0753  2
;   647          0754  2 LOCAL
;   648          0755  2     cr_lf : WORD INITIAL (%X'0D0A'),                    ! Local CRLR datum.
;   649          0756  2     mode : BYTE INITIAL (prt$c_uw);                     ! Hardware Protection code for User Writeabl
;   650          0757  2
;   651          0758  2 MAP
;   652          0759  2     desc : REF $BBLOCK;                                 ! Signal message descriptor.
;   653          0760  2
;   654          0761  2 IF PROBEW (mode, desc [dsc$w_length], .desc [dsc$a_pointer])     ! Can a CRLF be appended to the string?
;   655          0762  2 THEN                                                    ! Yes, append the carriage return line feed
;   656          0763  3     BEGIN                                               ! onto the line and adjust the length.
;   657          0764  3     .desc [dsc$a_pointer] + .desc [dsc$w_length] = .cr_lf;
;   658          0765  3     desc [dsc$w_length] = .desc [dsc$w_length] + 2;
;   659          0766  2     END;
;   660          0767  2
;   661          0768  2 bad$sta_io (IO$_WRITELBLK, .desc);                      ! Display the error message.
;   662          0769  2
;   663          0770  2 RETURN FALSE;                                           ! Inhibit $PUTMSG from duplicating the effor
;   664          0771  2
;   665          0772  1 END;    ! of GLOBAL ROUTINE bad$putmsg_actrtn


                                    0004 00000      .ENTRY  BAD$PUTMSG_ACTRTN, Save R2                              ;  0730
                          52  0D0A    8F  B0 00002  MOVW    #3338, CR_LF                                            ;  0750
                          51    04    90  B0 00007  MOVB    #4, MODE                                                ;
                          50    04    AC  D0 0000A  MOVL    DESC, R0                                                ;  0761
                    04    B0    60    51  0D 0000E  PROBEW  MODE, (R0), a4(R0)                                      ;
                                      0D  13 00013  BEQL    1$                                                      ;
                          51    60    3C  00015     MOVZWL  (R0), R1                                                ;  0764
                          51    04    A0  C0 00018  ADDL2   4(R0), R1                                               ;
                          61    52    3C  0001C     MOVZWL  CR_LF, (R1)                                             ;
```

```
                                60          02  A0 0001F          ADDW2   #2, (R0)                                ; 0765
                                            50  DD 00022 1$:      PUSHL   R0                                      ; 0768
                                            20  DD 00024          PUSHL   #32                                     ;
                             FEE3   CF      02  FB 00026          CALLS   #2, BAD$STA_IO                          ;
                                            50  D4 0002B          CLRL    R0                                      ; 0770
                                                04 0002D          RET                                             ; 0772

; Routine Size:  46 bytes,    Routine Base:  $CODE$ + 030D

;  666          0773  1
```

```
;  668          0774  1 ROUTINE bad$bugcheck : JSB NOVALUE =
;  669          0775  1 !++
;  670          0776  1 !
;  671          0777  1 ! Functional Description:
;  672          0778  1 !
;  673          0779  1 !    This routine is written over EXE$BUG_CHECK, in order to trap any
;  674          0780  1 !    bugcheck that might occur during execution of Stand Alone BAD.
;  675          0781  1 !
;  676          0782  1 !    EXE$BUG_CHECK is overwritten with a 'private' routine, because the
;  677          0783  1 !    console media containing the executive is not resident, hence the
;  678          0784  1 !    bugcheck code cannot be loaded.
;  679          0785  1 !
;  680          0786  1 !--
;  681          0787  2 BEGIN
;  682          0788  2
;  683          0789  2 BUILTIN
;  684          0790  2     HALT,
;  685          0791  2     MFPR,
;  686          0792  2     MTPR;
;  687          0793  2
;  688          0794  2 LOCAL
;  689          0795  2     msg_ptr : REF VECTOR [, BYTE],
;  690          0796  2     char;
;  691          0797  2
;  692          0798  2 MTPR (%REF (ipl$_power), pr$_ipl);                    ! No interrupts allowed!
;  693          0799  2 CON$OWNCTY();                                         ! "Allocate" console terminal
;  694          0800  2 msg_ptr = bugcheck_msg;                               ! Point to the message text.
;  695          0801  2
;  696          0802  2 DECR index FROM 14 TO 1 DO                           ! Set up for synchronous IO
;  697          0803  3     BEGIN                                            ! of the 'Bugcheck' message.
;  698          0804  3     char = .msg_ptr[0];  msg_ptr = .msg_ptr + 1;     ! Get next character
;  699          0805  3     CON$PUTCHAR(.char);                              ! Output character
;  700          0806  2     END;
;  701          0807  2
;  702          0808  2 HALT ();                                             ! Stop the processor!
;  703          0809  1 END;                                                 ! Note that the message text is appended to

                                12            1F  DA 00000 BAD$BUGCHECK:
                                                           MTPR    #31, #18                                        ; 0798
                            00000000G         00  16 00003           JSB     CON$OWNCTY                          ; 0799
                            52      0000V     CF  9E 00009           MOVAB   BUGCHECK_MSG, MSG_PTR               ; 0800
                            51                0E  D0 0000E           MOVL    #14, INDEX                          ; 0802
                            50                82  9A 00011 1$:       MOVZBL  (MSG_PTR)+, CHAR                    ; 0804
                            00000000G         00  16 00014           JSB     CON$PUTCHAR                         ; 0805
                            F4                51  F5 0001A           SOBGTR  INDEX, 1$                           ; 0802
                                              00  0001D           HALT                                        ; 0808
                                              05  0001E           RSB                                         ; 0809

; Routine Size:  31 bytes,    Routine Base:  $CODE$ + 0338

;  704          0810  1 OWN                                                 ! this routine via PSECT manipulation.
;  705          0811  1     bugcheck_msg        : PSECT ($CODE$) VECTOR [14, BYTE]
```

```
;  706          0812  1                              INITIAL (BYTE (%CHAR (%O'015', %O'012', 0, 0),
;  707          0813  1                                             'Bugcheck'
;  708          0814  1                                             %CHAR (%O'015', %O'012'))),
;  709          0815  1      bad$bugcheck_end    : PSECT ($CODE$) VECTOR [0];
;  710          0816  1 !
;  711          0817  1 ! END of ROUTINE bad$bugcheck
;  712          0818  1 !
;  713          0819  1
```

```
;  715          0820  1 ROUTINE bad$install_code =
;  716          0821  1 !++
;  717          0822  1 !
;  718          0823  1 ! Functional Description:
;  719          0824  1 !
;  720          0825  1 !    This routine is called to install KERNEL mode code.  We call it
;  721          0826  1 !    to overwrite EXE$BUG_CHECK with local code (bad$bugcheck).
;  722          0827  1 !
;  723          0828  1 ! Implicit Inputs:
;  724          0829  1 !
;  725          0830  1 !    Code and Data resident between 'bad$bugcheck' and 'bad$bugcheck_end'.
;  726          0831  1 !
;  727          0832  1 ! Implicit Outputs:
;  728          0833  1 !
;  729          0834  1 !    EXE$BUG_CHECK has been overwitten with the code specified above.
;  730          0835  1 !
;  731          0836  1 ! Routine Value:
;  732          0837  1 !
;  733          0838  1 !    SS$_NORMAL
;  734          0839  1 !
;  735          0840  1 !--
;  736          0841  2 BEGIN
;  737          0842  2
;  738          0843  2 ini$writable ();                                        ! Make VMS kernel writeable.
;  739          0844  2 CH$MOVE (bad$bugcheck_end - bad$bugcheck, bad$bugcheck, exe$bug_check);
;  740          0845  2 ini$rdonly ();                                          ! Reset VMS kernel back to read only.
;  741          0846  2
;  742          0847  2 RETURN ss$_normal;
;  743          0848  2
;  744          0849  1 END;    ! of ROUTINE bad$install_code


                                                        0035A           .BLKB   2
                                                00  00  0A  0D  0035C BUGCHECK_MSG:
                                                                    .ASCII  <13><10><0><0>                          ;
                                    6B  63  65  68  63  67  75  42  00360       .ASCII  \Bugcheck\                  ;
                                                    0A  0D  00368   .ASCII  <13><10>                                ;
                                                            0036A   .BLKB   2
                                                            0036C BAD$BUGCHECK_END:
                                                                    .BLKB   0


                                                            0FFC 00000 BAD$INSTALL_CODE:
                                                                    .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11    ; 0820
                                        00000000G   00  16  00002   JSB     INI$WRITABLE                            ; 0843
                    00000000G   00      C3  AF          31  28  00008   MOVC3   #49, BAD$BUGCHECK, EXE$BUG_CHECK  ; 0844
                                        00000000G   00  16  00011   JSB     INI$RDONLY                              ; 0845
                                    50              01  D0  00017   MOVL    #1, R0                                  ; 0847
                                                        04  0001A   RET                                             ; 0849

; Routine Size:  27 bytes,    Routine Base:  $CODE$ + 036C

;  745          0850  1
;  746          0851  1
```

```
;  747          0852  1 END     ! of MODULE badstasys
;  748          0853  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                                PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $OWN$                          16  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                         56  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                        903  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                              Library Statistics
;
;                                      -------- Symbols --------      Pages       Processing
;       File                           Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1        18619        37         0        1000        00:01.5



;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:BADSTASYS/OBJ=OBJ$:BADSTASYS MSRC$:BADSTASYS/UPDATE=(ENH$:BADSTASYS)

; Size:         885 code + 90 data bytes
; Run Time:         00:14.3
; Elapsed Time:     00:58.3
; Lines/CPU Min:     3591
; Lexemes/CPU-Min: 26593
; Memory Used:  128 pages
; Compilation Complete
```

BADREPORT
LIS

BADBLK

SCANFILE
LIS

BADBLOCK
MAP

BADMAIN
LIS

BADIO
LIS

GETREQ
LIS

BADMSG
LIS

BADSTATUS
LIS